The Role of Architecting in Systems of Systems

Peter Brook
Dashwood Consulting
Dashwood House, Manby Rd
Malvern, Worcs WR14 3BB
UK

dashwoodbrook@aol.com

ABSTRACT

Architecture and architecting jointly lie at the heart of all activity to develop and operate Systems of Systems (SoS). This paper reviews this growing field in relation to complex systems generally, and then discusses those aspects which are specific to SoS, including the use of architectural frameworks. The aim is to lay some foundations for the talks which follow.

1.0 INTRODUCTION

1.1 Architecture and Architecting

From the earliest times, mankind has used the practice of architecture to describe how it has designed and built its most important creations. For the last few millennia and until recent times, the subject was associated exclusively with large physical constructions – for example pyramids, public monuments, and religious, civic and private buildings. That meaning continues today, but as we have extended our capability to build more and more complex entities, the nature and scope of architecture has had to expand accordingly. The mid-20th Century saw the emergence of systems engineering as the all-embracing discipline, with architecting at its centre, for the design and building of a new class of more complex products and systems (e.g. cars, ships and aeroplanes) which incorporated software and digital control. In parallel, a range of similar tools and methods emerged for designing information systems, i.e. those comprising 'just' software running on computing platforms over networks. The growth in how architecting is understood and applied continues today and is accelerating rapidly as the technical, commercial, military and security communities learn how to grapple with the new 21st Century systems challenges characterised by ubiquitous computing and near-global connectivity. And architecting is now being applied to the design of such diverse and complex entities as robotic swarms, large organisations and digital cities.

The most widely referenced work on the subject of architecting, and still a good introductory text, is that of Rechtin and Maier [1]. For a more up-to-date treatment, see Sillitto [2], who addresses some of the same themes as are covered here.

As our starting point, we take two widely used modern definitions for architecture:

- *ISO 42101 [3]: The fundamental concepts or properties of a system in its environment embodied in its elements, relationships and the principles of its design and evolution.*
- *OMG (Object Management Group): The organisational structure and associated behaviour of a system. An architecture can be recursively decomposed into parts that interact through interfaces and that connect parts and constraints for assembling parts.*

In everyday language we use the term *architecture* to describe both the artefacts – i.e. models, pictures and designs for systems - and the generic discipline of drawing, assessing and using such artefacts (as in the term: *A School of Architecture*). The technical community now uses the word architecture only for the former meaning, usually preferring the word *architecting* to describe the discipline, along with the associated concepts, principles, frameworks, heuristics, tools and methods. This usage is adopted here.

1.2 Types of Architecture and Architecting

Some idea of the breadth of the subject of architecting can be gained from Figure 1, which shows the range of things we now consider as having some form of architecture (this is based on a treatment by MIT [4]).

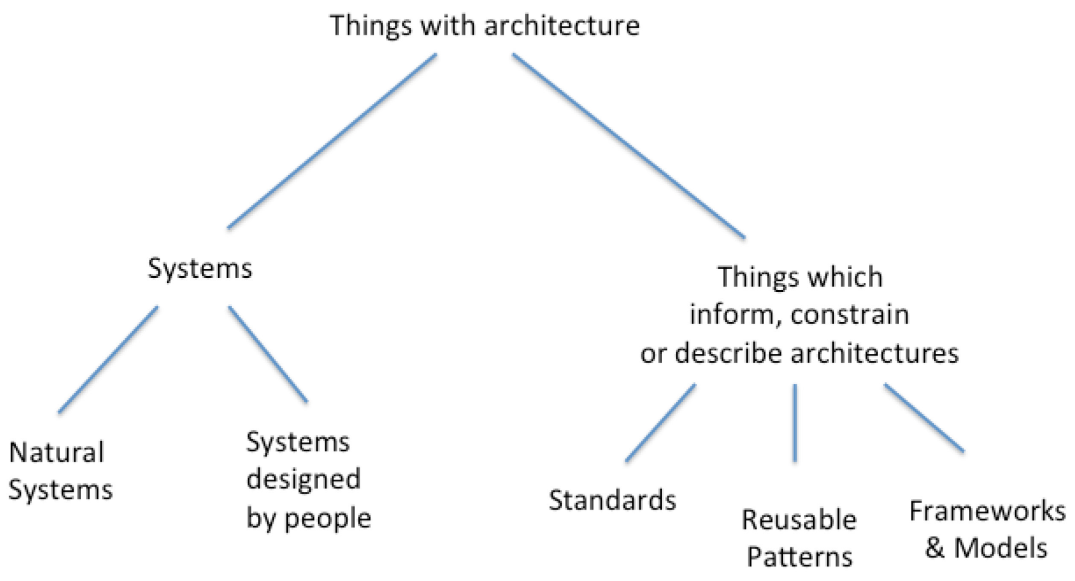

Figure 1: Things with Architecture.

We see things with architecture falling into two main classes:

- *Systems:* which fall into two further classes: those which occur in nature and those we build; and
- *Things which inform, constrain, or describe systems,* which comprise standards, re-usable patterns and frameworks & models.

The architecture of natural systems will not be covered here, although interestingly some authors are now turning to models based on living organisms to describe complex man-made systems and organisations [5]. Standards, patterns and frameworks all feature strongly in the architecting of SoS.

Architecting now covers a range of practices which depend on the types of systems we build. Table 1 shows the leading types of architecture, and the type of system to which they apply. Practices in all these fields are rapidly developing, merging, and borrowing from each other. This trend is driven especially by the growing convergence of physical and distributed IT systems – the so-called *cyber-physical systems*. SoS Architecting draws on all these types to a greater or lesser degree.

Table 1: Systems Types and Associated Architectures.

Type of System	Associated Architectures	Features
General Systems: Systems comprised of an arbitrary mix of hardware and software, such as cars, aeroplanes, consumer electronics.	**Functional Architecture**: a logical view of the solution, creating a bridge between user needs and system design (sometimes embodied in system requirements). **High Level Design**: system-wide features of the solution, expressed in terms of *Behaviour, Structure* and *Layout.*	The practices are brought together with other processes in *Traditional Systems Engineering* (see for example INCOSE Handbook). Modelling tools increasingly based on SysML.
IT Systems: Systems solely concerned with the processing of information.	**Business Architecture:** how the organisation intends to use the information, including associated business or operational processes. **Information Architecture:** exchange, storage, ownership of information; information types, conversions and transformations. **Systems Architecture:** logical (functional) view of operating software. **Technical Architecture:** typically computing and network resources on which the system will depend.	Many individual practices, often specific to domain of use. Modelling based on UML and variants.
Enterprise Systems: IT systems explicitly designed to support organisations.	**Enterprise Architecture:** - Incorporates IT Systems Architecture (as above). - Legacy, Transitional and Goal Architectures. - Frameworks and reference models.	Specifically addresses linkage with organisational strategy and business change. Generic practices codified in TOGAF (see below).
Service Systems: Systems (mainly IT) which provide services to each other or an organisation.	**Service Oriented Architecture:** a design pattern based on distinct software applications providing services to others.	Prevalent in business IT and EA as a way of providing flexibility, ease of reuse and vendor independence. Emerging standards codified in SOA Framework.

1.3 Why Do We Do Architecting?

The motivation for architecting, and the benefits for doing so, both depend on the context in which it is applied. The following are quite generic, and apply across all areas of application – including that of Systems of Systems:

- To allow us to relate the overall properties of a potential system, including its emergent behaviour, to its requirements, expected benefits, costs and achievable timescales (this is usually done through use of models).

- To allow us to evaluate alternative solution options before full commitment to the relatively more expensive process of building (using decision support tools).
- To allow all parties to identify the scope and boundary of the system to be built, and explore its interactions with others in its environment; these include physical interfaces and those involved in the exchange of information, services and energy across the system boundary (using for example architectural frameworks).
- Following the last, to allow us to relate solutions to the businesses or operations they are intended to support, for example to evaluate the implications for business process change, or new military missions and associated tactics (using mission or business process modelling).
- To allow us to break a system down into elements, which can be built or bought, by defining their desirable features and the interfaces between them. This may include identifying points at which agreed standards should be applied internally to the system, or standard patterns applied (through recursive application of models).
- To allow us to allocate system functions to hardware, software and people, and understand the interactions between them.
- To allow essential non-functional aspects of a solution – principally safety, security and resilience – to be evaluated, and conformance with customer needs or regulatory standards assessed (using specific modelling tools).
- To allow us to set out a strategy for progressive integration and testing of the built system, while still at the design stage (through test planning set against the architectural elements).
- If aesthetics are important, to be able to show the overall appearance of a system to intended users, sponsors, planners for feedback and/or approval (by rendering the external look of the physical solution).

2.0 ARCHITECTING SYSTEMS OF SYSTEMS

2.1 The Nature of SoS

Here we turn to the issues concerned with applying architectures and architecting to SoS. At the current state of the art, it is fair to say that we are still extrapolating best practice from the domain of engineering of individual systems into the more complex and fast-moving space occupied by SoS. We can only speculate as to whether a new discipline might emerge in time to cover SoS. Before proceeding further it is important to address two related, underlying questions:

- *'If a system of systems is also a system, why do we need anything different from normal systems engineering?'; and*
- *'Aren't most systems also systems of systems, since they comprise sub-systems which are also systems?'*

These seemingly simple questions can still give rise to robust debate, for example on Internet sites. In this author's view, the answer lies in the viewpoint taken – specifically whether one is arguing *systemically* (interested in how the system behaves as a whole and how it interacts with its environment) or *systematically* (interested in how a system is made up, how the parts are built, put together and interact with each other).

Using this distinction, we can say:

- From a *systemic* point of view, a SoS is 'simply' another system while the constituent systems are working together. In keeping with all systems, they may be classified into a number of types, for example:

    - *Closed and directed*, built as a whole, with relatively stable and long-lasting interfaces with the wider world, with behaviour fairly possible to predict (for example *Air Traffic Control)*;
    - *Semi-open,* largely built and with some control, but with fast-moving technology, including autonomous digital systems and human involvement, and with behaviour consequently difficult to predict (for example *Road Traffic Systems*); and
    - *Totally open, little of no control,* evolving organically as a result of internal forces (for example, international *Stock Exchanges* or *Social Systems*).

- From a *systematic* point of view, the differences between systems and SoS are major and deep-seated. The constituent systems of a SoS possess independent viability, and the ways in which they are developed – for example under different management – are not typically found in the world of building self-standing systems. (It follows that the SoS Types identified by DoD [6] are essentially systematic rather than systemic in nature.)

Our concern in the rest of this paper is with issues lying in the systematic domain, i.e. how SoS are put together for specific purposes, and the role which architecting has to play.

2.2 General Principles for Architecting SoS

The concepts developed for architecting individual systems all find their place in some form or other in the more complex world of SoS. But there are specific practices – or modifications to established usage – which arise specifically as a result of those issues which arise in the SoS domain, and therefore need to be addressed.

We find that it is not always possible to define a single, all-encompassing and enduring architecture which can be developed once and used for the duration of the SoS. As we will describe below, we may have to work in a more pragmatic and agile manner, which requires change to normal engineering management practices.

The following are some of the issues which come into play and drive the need for specific approaches:

- *Operational independence of systems*: systems can come together to form a SoS from different organisations and organisational cultures.
- *Asynchronous life cycles:* constituent systems may be built and maintained in different timescales, i.e. starting at different points in time with varying duration, sometimes dictated by the novelty of the underlying technology and the time it takes for it to be matured to the required level.
- *Technical diversity:* differing architectures, patterns and standards among the constituent systems*,* not always fully recorded or made known to those responsible for integrating the SoS.
- *Unexpected emergent behaviour:* as systems join the SoS – and leave it.
- *Uncertainty about the future:* SoS are often required to operate in a variety of different configurations, for example joining up different commercial partners or military allies, or with novel systems, or operational missions not all of which will have been foreseen (or might even have been foreseeable) at the point at which the constituent systems were designed and built.

Much of the overall response to these challenges lies outside the purely technical domain, for example requiring significant changes to management processes or even how the whole organisation (or enterprise) approaches systems development and operational use. These wider issues will be covered in other papers in this volume. For the rest of this paper we confine ourselves to the role which architecting has to play as part of this wider response.

The following general principles have been found to apply (see for example Sheard [7] or Brook et al. [8]):

- Where one has control over the development of constituent systems, aim for *loose coupling* between them; this implies that systems can be altered without the alteration necessarily affecting others. Taken to extreme, this can sometimes be viewed as producing the ability to "plug and play", where systems are so independent that they can be changed without affecting the overall SoS behaviour. That extreme is considered an unlikely nirvana at the present time.
- Aim for high degrees of *modularity,* with systems boundaries aligned to generic functionality, commercial standards and market-leading component systems wherever possible. This not only makes the individual systems easier to build and maintain, but encourages interoperability and eases the difficulty of modifying parts of the SoS should this become necessary to create coherence of the whole. The use of a common Technical Reference Model is a well-accepted way of achieving this.
- Create clear *architectural separation* between the more volatile fast-moving SoS elements and the more stable and long-lasting ones, which are typically infrastructure. In IT this means procuring communications and common support as a service, while allowing more agile and mediated approaches at the applications levels. Physical systems – for example military tanks, warships and aircraft, and commercial automotives – design the physical platform in such a way as to allow the incorporation of multiple generations of the electronic and computing subsystems over system or product lifetime. This allows adaption to later changes in context of use, including market forces.
- Design the constituent systems for *composability*, allowing them to be put together in the most number of operational configurations, largely as a response to future uncertainty [9]. Amongst other approaches, this involves investigating the performance of the SoS in a number of possible missions and therefore technical configurations while still at the design stage. (Other approaches in this list – such as modularity and architectural separation – also contribute to greater composability.)
- Pay attention to interfaces; these must be pinned down and agreed – technically and contractually – and rigourously tested.
- Write down and re-use successful design patterns. The systems we build are rarely completely novel and design patterns embody industry-wide (or organisation-wide) knowledge of what works. They also assist with encouraging use of standards for the reasons just given.
- Wherever possible, model before build. Frameworks have a part to play in this and are covered extensively below, but other sorts of models may well be necessary, especially to evaluate dynamic performance – or even look and feel.
- A number of engineering and project management strategies are also important and discussed elsewhere in this volume, comprising some or all of the following:
  - Organising groups of projects to proceed in tranches or epochs, delivering enhanced SoS capability in stages, with architectures converging over time.
  - Establishing and maintaining high degrees of communication – informal and informal – between architects and designers across relevant projects.
  - Use of supra-project management structures, using recognised Programme and Portfolio Management process, with the possible appointment of Lead System Integrators.

2.3 Broad Application of Architecting to SoS Types

We can gain further insight into the various ways in which architecture enters into the technical and engineering management of SoS by setting out the types in the manner shown in Figure 2.

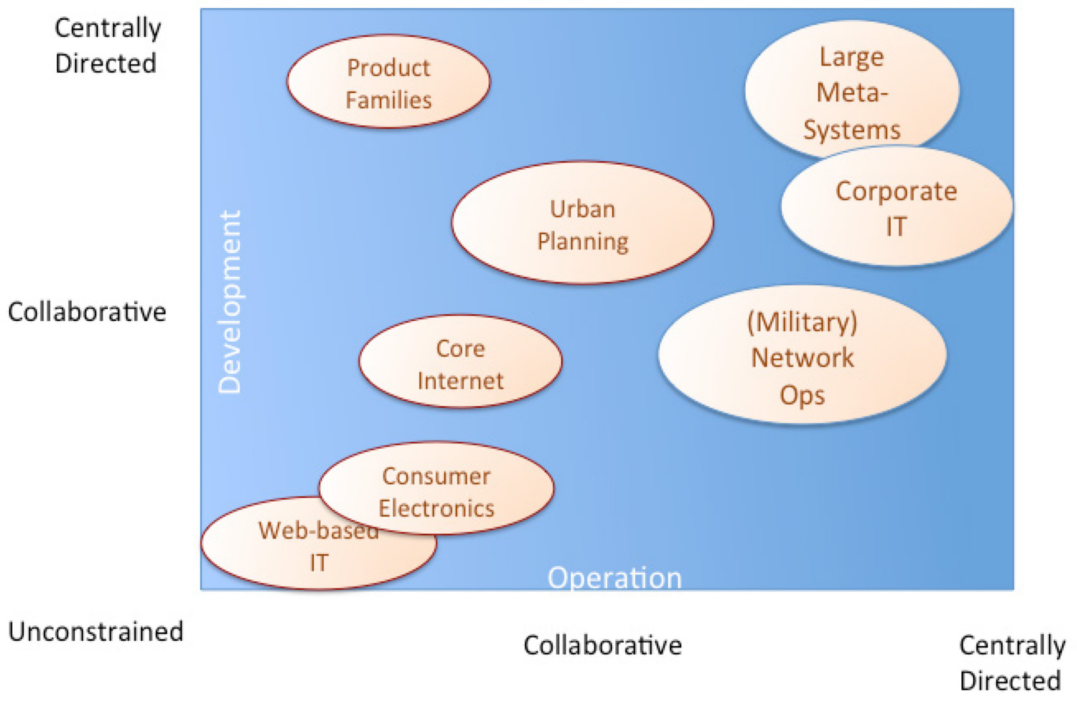

Figure 2: Some Systems of Systems Types.

The axes of the figure relate to how the SoS are managed in *Development* (vertical axis) and *Operation* (horizontal axis). At the common origin we have systems which are not managed in any collective way, and as we move along each axis, we depict increasing amounts of management being applied amongst the parties concerned. At the extreme ends (bottom right and top left) we place systems which are managed in a highly directed manner, and along the axes increasing amounts of direction.

[The treatment borrows heavily from that of Ref [6]: *Unconstrained* here corresponds to DoD's *Virtual*; *Collaborative* here covers both DoD's *Acknowledged* and *Collaborative*; and *Directed* means the same in both models.]

Superimposed on this framework are a number of important SoS types. The breadth of the ovals indicates that this treatment is somewhat approximate, but it does allow us to distinguish a number of major architectural approaches, each of which incorporates to some degree the general principles of the last section.

For each of the types shown it is possible to say something about the architectural approach:

- *Web-based IT*: characterised by applications and associated standards that emerge from the community, with the best being selected by adoption by the worldwide community users in a Darwinian manner.
- *Consumer electronics*: we include here all the things we buy and assemble to create the environment for computing, gaming, home entertainment, audio photography/video – connecting us while at home and on the move. Suppliers simultaneously collaborate and compete to create market standards. The architectural principle is *plug and play*, leaving it to the consumer to integrate, install and maintain his/her own SoS capability, often with no underlying architecture. Standards are developed either by large companies, where they have the industrial power, but more usually in trade associations and industry groupings, acting in mutual self-interest to stimulate the resulting market. [The move towards a so-called *Internet of Things* is a further extension of the same idea.]

- *Core Internet:* essentially the technical and operational architecture of the internet, developed to agreed architectural standards by the Internet Engineering Task Force, a voluntary body of commercial providers, academics, on top of this we have the World Wide Consortium covering HTML and XML. These and other such bodies operate on a voluntary basis, with their products available to service and product providers to develop infrastructure to greater or lesser degrees of collaboration.
- *Product Families:* now becoming the norm in major commercial industries such as automotive, aircraft, cameras, office equipment, mobile phones, etc. Here a variety of products are developed with many common elements, motivated by the need to reduce development and operating/maintenance costs, while offering a degree of variety to the consumer. Development is typically closely directed across multiple products which subsequently operate autonomously, except in so far as they might share logistics/ repair/maintenance. The products are bound together through a *Common Platform Architecture*s comprising key elements of underlying common physical and software components, interfaces, manufacturing and test environments. Military systems procurement is increasingly adopting similar approaches for warships, tanks and air platforms.
- *Product Families:* now becoming the norm in major commercial industries such as automotive, aircraft, cameras, office equipment, mobile phones, etc. Here a variety of products are developed with many common elements, motivated by the need to reduce development and operating/maintenance costs, while offering a degree of variety to the consumer. Development is typically closely directed across multiple products which subsequently operate autonomously, except in so far as they might share logistics/ repair/maintenance. The products are bound together through a *Common Platform Architecture*s comprising key elements of underlying common physical and software components, interfaces, manufacturing and test environments. Military systems procurement is increasingly adopting similar approaches for warships, tanks and air platforms.
- *Large Meta-Systems:* for example trans-national Air Traffic Control, or certain wide-area military systems such as Air Defence. Here we have a mix of platforms, sensors, communications and control systems, forming a top-down integrated programme to meet agreed objectives - typically managed by a lead organisation such as a Lead System Integrator. The architecture may combine elements of the General Systems approach to architecture shown in Table 1, i.e. using Functional Architecture and High-Level Design for allocating performance to constituent systems and Information Architecture for internal and external information exchange.
- *Corporate IT:* here we are typically interested in bringing together, rationalising, merging or updating IT systems spanning multiple departments of an organisation, most likely geographically separated. This is the area for which *Enterprise Architecture* was designed and is very widely applied, supported by TOGAF.
- *Military Networks Operations:* single-nation systems increasingly apply the *Architectural Frameworks* described below, with extensions to conventional systems engineering to include mission analysis. Most nations will also build some form of underlying communications infrastructure, procured and supported by a single agency. Allied interoperability brings involves a wider set of issues; basically some form of collaborative approach such as Alliance-wide standardisation.
- *Urban Planning:* Cities of the future (and some of the present) use architectural principles to develop and maintain an overall pattern, which drives common infrastructure in the form of roads and public utilities, while allowing the community to evolve in response to changes in social need, culture, environmental targets, demographics and technology. Here a government body will lay down guidelines, plans and key standards, and scrutinise submissions for new developments for compliance, otherwise leaving scope for maximum variety. The plan also changes over time, but on

longer timescales then constituent developments. This form of SoS is expected to dominate many of our lives in the future.

3.0 ARCHITECTURAL FRAMEWORKS

3.1 Background

Much of the practical work on architecting SoS is based on the use of *Architectural Frameworks,* the leading examples of which will be described in this section.

A good working definitions of an architectural framework is as follows:

> *A skeletal structure that defines suggested architectural artefacts, describes how these artefacts relate to each other, and provides generic definitions for what they might look like.*

Frameworks provide a mechanism to support a shared understanding and comparison of architectural information, in a way which is independent of individual projects or organisations, and expressed at a level of abstraction which sits above specific designs. This becomes especially important in architecting SoS, which involves at its core the bringing together of multiple organisations and systems; in this arena, harmonising how architectures are described has understandably had a major impact. Specific domains and industries have adopted variations on the same underlying concepts which best suit their type of business or system. (Session provides a good review of the main types of framework.

The origin of such frameworks lies in the pioneering work of Zachman [10], who developed a generic model and associated concepts of use which are still evident in more modern versions. His motivation was encouraging businesses to adopt a more disciplined approach to managing the design and implementation of its information systems. His work was taken up by the US DoD in the form of TAFIM (Technical Architectural Framework for Information Management), which was in turn extended to produce the FEAF (Federal Enterprise Architecture Framework), intended to use across US Government and its agencies.

TAFIM was picked up and further developed by the Open Group to form TOGAF [11] (*The Open Group Architecture Framework*), now the acknowledged world standard for Enterprise Architecture in the civilian business world, and increasingly finding a place in parts of the military systems domain.

Within the DoD, TAFIM went through several iterations to become the *C4ISR Architecture Framework* and finally DoDAF *(DoD Architectural Framework),* which is in widespread use today. In parallel, the UK MOD produced MODAF (MOD *Architectural Framework*) around 10 years ago, by adding features to DoDAF. This in turn has formed the basis for the NAF (*NATO Architectural Framework),* while other nations have produced their own similar frameworks. Moves are now under way to unite the above variants: the UPDM (Unified Profile for DoDAF and MODAF) Program, supported by the Object Management Group, is bringing together the two leading military frameworks; further convergence is also under way between UPDM, NAF and other national standards.

As things stand, there are two main strands in architectural frameworks: those originating from the military, such as DoDAF, MODAF, NAF, etc.; and the leading commercial framework, TOGAF [12]. Both have relevance to SoS and their main features are now described, with an indication as to how they are applied.

3.2 Military Architectural Frameworks

Although a number of variations exist, all the leading military architectural frameworks (MODAF, DoDAF, NAF, etc.) share a common set of objectives: *to provide a specification of how to represent an integrated model of an enterprise or mission, linking the operational aspects to the systems or services which provide the required capability, with appropriate standards and programmatic aspects.*

They achieve this is by setting out in a formal manner a number of views, which may be selected for the task in hand. These views are organised into a number of types (or viewpoints), as shown in Figure 3 [13], which is based on MODAF 1.2.

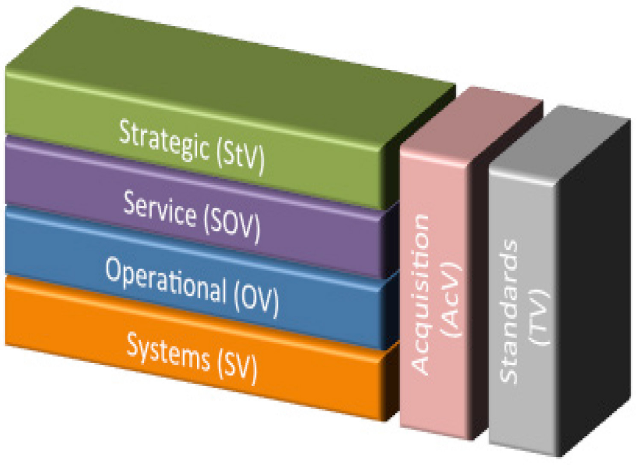

Figure 3: Summary of MODAF Views.

The content of each of the types of MODAF Architectural viewpoints is summarised in Table 2. Each viewpoint comprises a number of views, and selectivity is required to choose those most relevant to the task in hand. This might be left to the judgement of the technical team, or more likely laid out by the organisation or those responsible for integrating the SoS of which the system of interest forms a part. This is to ensure consistency in the use of views across the SoS, as an aid to across-project understanding and exchange between architects. (It also helps in populating a repository if one is in use.)

Table 2: MODAF Architectural Viewpoints and Indicative Content (based on MODAF 1.2).

Architectural Viewpoint	Indicative Content
Strategic Viewpoint (StV) - 6 views	The desired business/operational outcome, and the capabilities required to achieve it. Boundaries, vision and enterprise goals.
Operational Viewpoint (OV) - 11 views	How the operational solution is structured, expressed logically, i.e. in terms of functions performed by the constituent systems, with no prescribed solutions.
Service Oriented Viewpoint (SOV) - 7 views	Services exchanged between units in the operational views. Expressed in terms of service interactions and interfaces – again solution-independent.
Systems Viewpoint (SV) - 17 views	Description of the solution in physical terms (one or more solutions may be investigated). Shows how the solution systems interact to deliver the required capabilities.

Architectural Viewpoint	Indicative Content
Technical Viewpoint (TV) - 2 views	Description of applicable standards and how and where they are implemented in the solution(s).
Acquisition Viewpoint (AcV) - 2 views	Timelines of relevant projects, within and outside the boundaries of the solution option, to allow programmatic management and its relationship to technical maturity.
All Viewpoints (AV) - 2 views	A summary of the views selected and overall architectural context for the project. (Not shown in Figure 3.)

Figure 4 shows how some of the main MODAF views are related to each other in use. The strategic views show how capability is phased over time (with constituent project timelines as an additional option), with the Operational Views showing the relationship between the nodes of the solution in functional (logical) terms. The System Views show one or more solutions in terms of physical artefacts, software or people. Also shown are the trade-off's which take place between capabilities and scenarios at the higher two levels and between functions and solutions at the lower levels.

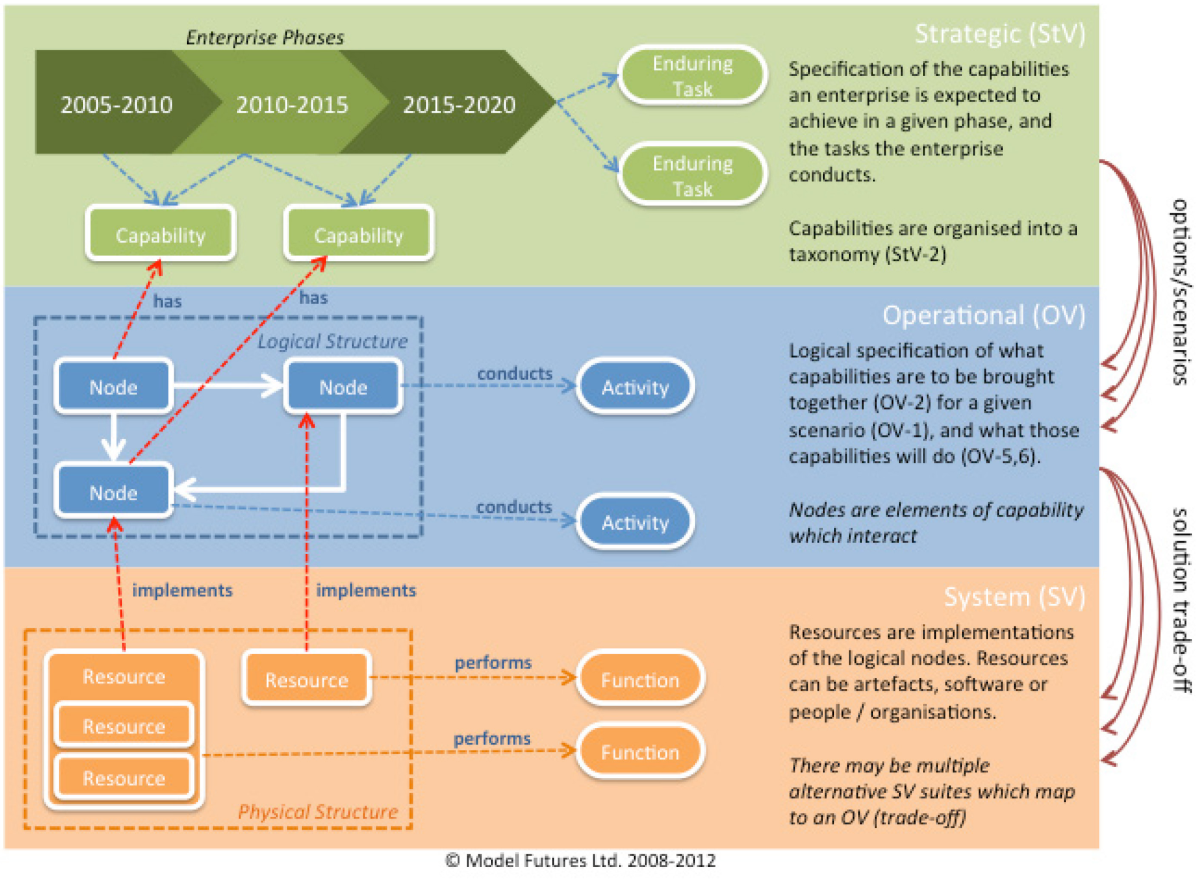
Enterprise Phases
2005-2010
2010-2015
2015-2020
Enduring Task
Enduring Task
Strategic (StV)
Specification of the capabilities an enterprise is expected to achieve in a given phase, and the tasks the enterprise conducts.
Capabilities are organised into a taxonomy (StV-2)
Capability
Capability
has
has
Logical Structure
Node
Node
Node
conducts
conducts
Activity
Activity
Operational (OV)
Logical specification of what capabilities are to be brought together (OV-2) for a given scenario (OV-1), and what those capabilities will do (OV-5,6).
Nodes are elements of capability which interact
options/scenarios
implements
implements
Resource
Resource
Resource
Resource
performs
performs
Function
Function
Physical Structure
System (SV)
Resources are implementations of the logical nodes. Resources can be artefacts, software or people / organisations.
There may be multiple alternative SV suites which map to an OV (trade-off)
solution trade-off
© Model Futures Ltd. 2008-2012

Figure 4: MODAF Usage – Schematic.

Further insight into Architectural Frameworks can be gained by looking at worked example of populated views in UPDM available on-line [14]. This takes for purposes of illustration a civilian Search and Rescue SoS.

MODAF does not define any particular process for its use, leaving architects free to implement whatever process standard the organisation or project chooses. There are clear advantages, in terms of allowing flexibility of use, but a corresponding disadvantage where architects are left in a process-free environment. The risk in such cases is poor project discipline and generation of views as an end in itself.

Figure 5 shows a recommended method for architecting, and the generation of associated information in terms of models and views, based on the treatment of Sillitto [2]. The model takes two main inputs as the starting point for architecting.

- *A statement of Purpose,* which might include project goals and measures of success, along with constraints (timescale, cost and operating environment). These could be expressed textually, for example, in the form of user requirements, or might be inherited by other systems in the SoS.
- Input Views (generated by users, or by advisors on their behalf) which diagram what the user is looking for, what he knows about the likely operational environment and mode of use of the system, along with an indication of needs to operate with other systems, services or other allied nations.

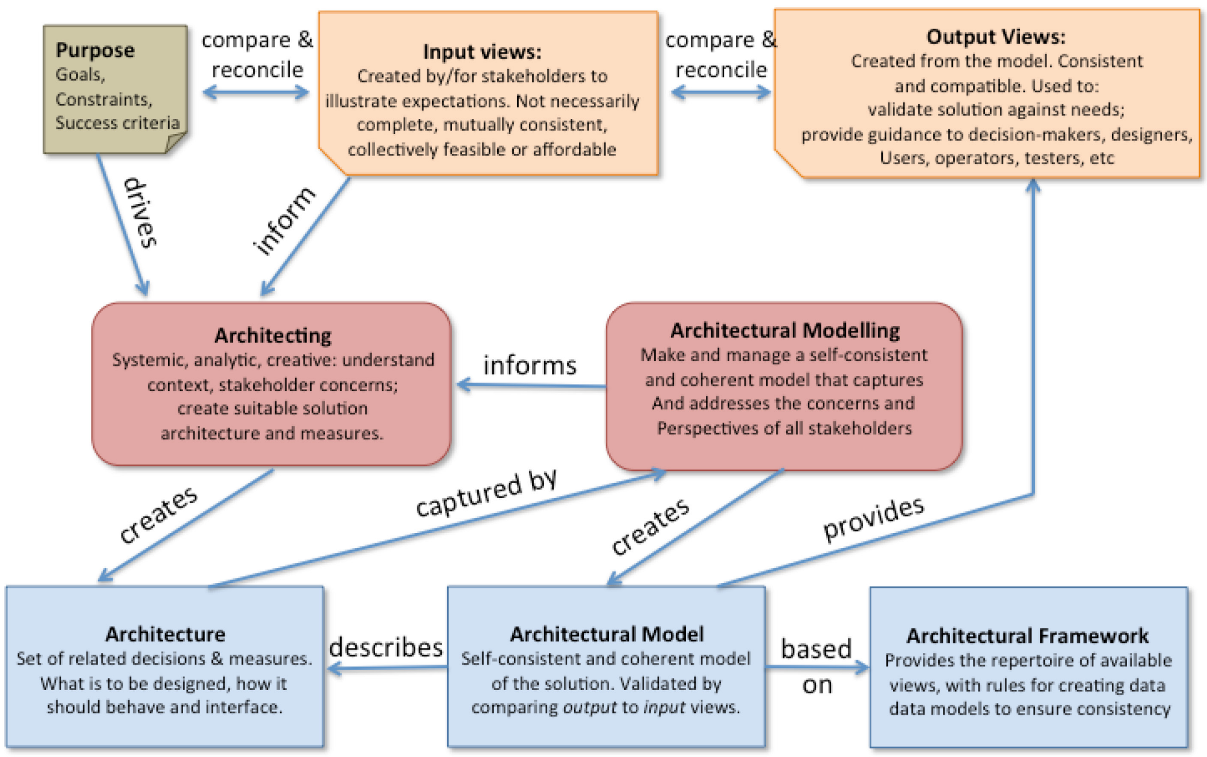
Purpose
Goals, Constraints, Success criteria
compare & reconcile
Input views:
Created by/for stakeholders to illustrate expectations. Not necessarily complete, mutually consistent, collectively feasible or affordable
compare & reconcile
Output Views:
Created from the model. Consistent and compatible. Used to: validate solution against needs; provide guidance to decision-makers, designers, Users, operators, testers, etc
drives
inform
Architecting
Systemic, analytic, creative: understand context, stakeholder concerns; create suitable solution architecture and measures.
informs
Architectural Modelling
Make and manage a self-consistent and coherent model that captures And addresses the concerns and Perspectives of all stakeholders
creates
captured by
creates
provides
Architecture
Set of related decisions & measures. What is to be designed, how it should behave and interface.
describes
Architectural Model
Self-consistent and coherent model of the solution. Validated by comparing *output* to *input* views.
based on
Architectural Framework
Provides the repertoire of available views, with rules for creating data data models to ensure consistency

Figure 5: A Model for Architecting with Frameworks - after Sillitto [2].

The middle layer of the model describes two main activities (which are not the same):

- *Architecting:* the creative process of thinking through systems issues, working with stakeholders and creating solutions, if necessary by reconciling conflicting views across stakeholders.
- *Architectural Modelling***:** the technical process of developing and maintaining self-consistent models of the system (or systems), usually involving a professional tool.

At the lowest layer of the model are the supporting artefacts:

- *The Architecture*: for example comprising a collection of documents, specifications, models, results of key decisions, and their rationale. There is a move towards expressing the entirety of the architecture in terms of models, dispensing with documents altogether. However, at the current state of the capability of tools we are obliged to work with a mix. (Some of the reasons for this are discussed in Section 3.4.)
- *An Architectural Model:* expressed as self-consistent set of views. Achieving self-consistency depends on the skill of the modeller, the choice of views, the power of the chosen tool and the quality of the underlying data model.
- *Architectural Framework:* providing the repertoire of views chosen for use in the architectural model, and the formal rules for binding them together.

The process is completed when the models created by the architectural modelling process are assessed for mutual consistency, validated against the stakeholders' views and the stated purpose, and any differences agreed by all concerned.

The approach process is quite general and can readily be applied iteratively: for example as part of an on-going life cycle or to a constituent system in a SoS. The organisation applying the process would be expected to conduct a technical review at key points in the architecture's evolution to ensure convergence towards an agreed solution with key decisions understood and bought into by all parties.

3.3 TOGAF

TOGAF is an open, internationally agreed and vendor-neutral method designed for undertaking Enterprise Architecture and IT-driven business change. It follows the general principles laid out by Zachman and shows how to link business strategy to IT implementation in an explicit manner.

It differs from the frameworks just discussed in a number of ways:

- It provides a generic life cycle process (the ADM - Architecture Development Method), shown in Figure 6;
- It contains artefacts for general use such as Technical Reference Models;
- It includes and rules on project governance, including criteria for technical and programme review; and
- It does not recommend any formal logic for development of views, leaving the matter to the market-leading tools, such as ArchiMate.

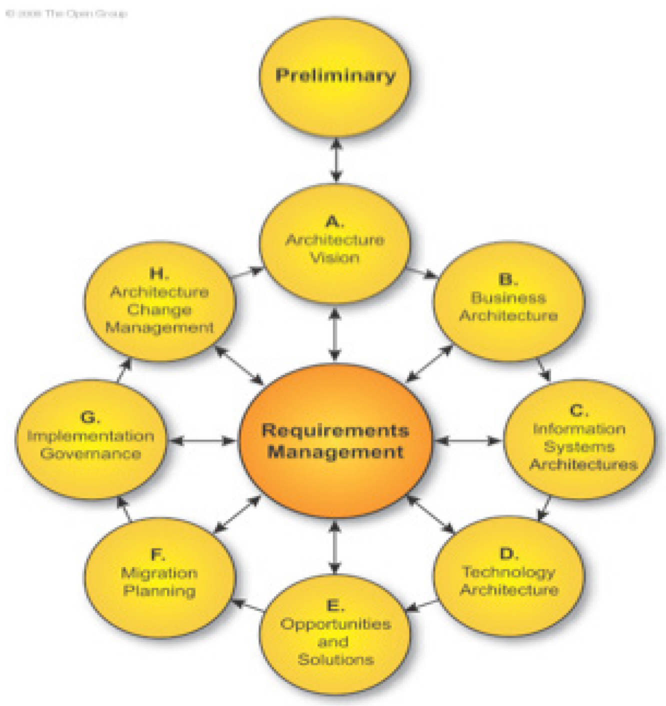
© 2006 The Open Group
Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Figure 6: TOGAF Architectural Development Method.

The ADM uses three types of architecture – *Business, Information* and *Technology* – which have to be modelling and linked using tools. It is divided into a number of stages – the main activities in each are summarised in Table 3. The model shown is of course a simplification; it recognises that in practice, some stages – especially the early ones, B to F – will have to be performed iteratively and repeatedly before arriving at compatible and affordable solutions, linked to the programme management environment.

Table 3: TOGAF Stages and Summary of Activities for Each.

TOGAF Stage	Summary of Activities
Preliminary	Obtain organisational support to undertake EA Establish team Define/customise tools, processes, etc.
A. Architecture Vision	Define business requirements Capture stakeholder concerns, constraints Develop value proposition, KPIs Identify risks and mitigation
B. Business Architecture	Develop business architecture Define organisational units, business functions and services to be covered Define Baseline (as-is) and Target (to-be) architectures

TOGAF Stage	Summary of Activities
C. Information Systems Architecture	Develop IS architecture, including data and applications Define Baseline (as-is) and Target (to-be) architectures Trace to Business Architecture (via modelling tools)
D. Technology Architecture	Develop technology architecture, including computing platforms and networks Define Baseline (as-is) and Target (to-be) Trace to Business Architecture (via modelling tools)
E. Opportunities and Solutions	Generate implementation and migration strategy Perform gap analysis propose solutions Produce roadmap showing evolution from baseline to target via intermediates stages Align roadmap to programmes and projects Confirm readiness for business transformation
F. Migration Planning	Sort projects into priority order De-risk delivery and migration Commence delivery of capability increment
G. Implementation Governance	Technical oversight and review Architectural contract between architects and projects, covering: choice of models, standards and interfaces
H Architecture Change Management	Determine whether to formally initiate a new architecture evolution cycle
Requirements	Develop and use a process for management of requirements throughout the ADM

A useful feature of the ADM is the way it addresses how the architecture is migrated from the *As-is* to the *To-Be Architectures*, if necessary via a number on *Transitional Architectures*. This is especially important when we have to build a SoS using legacy systems which need to be retired or updated. It also binds architectural governance into the project management structure via the *Architectural Contract.*

TOGAF also has a number of other important features, which those interested are encouraged to investigate for themselves (see [11]). These are contained in the so-called *Enterprise Continuum* which contains generic architectures in the form of re-usable patterns, recommended architecture descriptions and other artefacts relevant to the enterprise or the domain of use, for example transport, defence or generic IT.

3.4 Observations on Use of Frameworks

There is no doubt that Architectural Frameworks are bringing a degree of much-needed discipline to the architecting of SoS. They are currently well adapted to their intended purpose, which lies mainly in relating acquisition and management issues to high-level considerations of interoperability and information exchange.

However, they are still at a relatively immature stage of development, especially when compared with the full extent of the SoS Architecting problem. In particular [15]:

- AF's are currently weak in describing the physical (as opposed to information-centred) world.
- Some major concepts are semantically unclear, such as Capabilities and Services.
- They lack modelling mechanisms for major concerns such as human factors, safety, security and performance, which have to be modelled outside the frameworks.
- Most frameworks are 'stovepipes': closed, non-extensible and unable to work with each other; this leads on to lack of interoperability between tools.

In addition to the above, which are intrinsic to the frameworks themselves there are pitfalls to be avoided in using them in practical situations, which include:

- Poor use of patterns, treating each project or group of projects as a 'one-off'.
- Poor linkage between the viewpoints generated and other parts of the architecting discipline, such analysis of risk, cost and feasibility.

Perhaps most seriously, because of their origins in the technical domain and the need for specialist skills to operate the tools required to populate the views, there is a tendency to confuse drawing views with architecting in the broad. In the extreme we can find that senior decision-makers are removed from the relating and their use can be confidently expected to continue and develop for the indefinite future.

4.0 SUMMARY AND CONCLUSIONS

The emergence of Systems of Systems in the modern world, as a result of fast-moving trends in computing and communications, is creating major challenges for both those who design and build them, and the organisations which use them. This applies across all fields of human endeavour, including military affairs.

This paper has shown how conventional practices in the field of architecting are being extended to cover the additional issues which arise in developing SoS. Although much is now known, the nature and complexity of the subject area is growing fast, and one can expect further advances over the next 10-20 years, especially as the practice of architecting becomes more automated with tools of increasing capability, as part of the move towards *Model Based Systems Engineering* (MBSE).

5.0 REFERENCES

[1] E Rechtin and M Maier, *The Art of Systems Architecting*, CRC Press, 1996.

[2] H Sillitto, *Architecting Systems, Concepts, Principles and Practice*, College Publications, 2014.

[3] ISO 42010: Systems and Software Engineering – Architecture Descriptions. www.iso.org.

[4] Various authors: *The Influence of Architecture in Engineering Systems,* Engineering Systems Monograph, MIT, March 2004 (available online).

[5] R. Espejo and A. Reyes, *Organizational Systems; Managing complexity with the Viable Systems Model,* Heidleberg: Springer, 201.

[6] Dahmann, J. and K. Baldwin. *Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering.* IEEE Systems Conference, April 2008, Montreal, Canada.

[7] S A Sheard et al, *Principles of Complex Systems of Systems Engineering,* Systems Engineering, Vol. 12, No. 4, John Wiley Periodicals.

[8] P Brook and T Riley, *Enterprise Systems Engineering – Practical Challenges and Emerging Solutions,* INCOSE International Symposium, Rome, July 2012.

[9] H Sillitto, *Composable Capability – Principles strategies and methods for Capability Systems Engineering,* INCOSE International Symposium, Philadelphia, June 2013.

[10] J. A. Zachman, *A Framework for Information Systems Architecture,* IBM Systems Journal, vol. 26, no. 3, 1987.

[11] The Open Group Architectural Framework, www.opengroup.org.

[12] See R. Session, *A Comparison of the Top Four Enterprise-Architecture Methodologies*, Microsoft, May 2007. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb466232.aspx.

[13] Acknowledgement for Figs 3 & 4: Ian Bailey of Model Futures, http://www.modelfutures.com.

[14] DoD UPDM SAR Exemplar – Available online.

[15] Points made here are based in part on private communication with Dr Jean-Luc Garnier of Thales.